Deel 3

Tierelantijntjes met

Programmeren voor kinderen

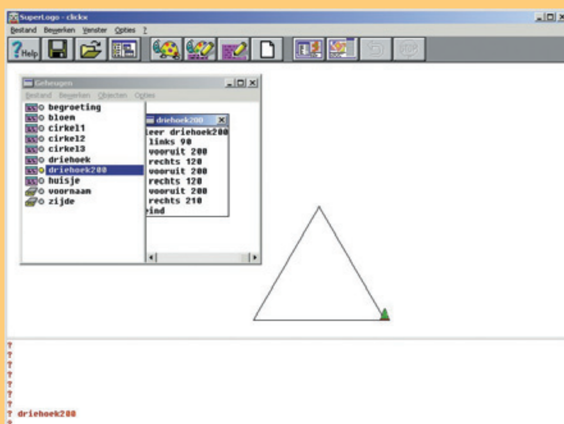
In de tweede aflevering hebben we je in ijltempo laten zien wat je met SuperLogo zoal kan aanvangen. We leerden ook al met variabelen werken. Laten we daar eens meer mee doen. En als toemaatje leren we je ook nog wat musiceren.

EVEN HET GEHEUGEN OPFRISSEN

Het basisscherm van SuperLogo is in drie grote delen verdeeld. Bovenaan zien we de titel-, menu- en knoppenbalken. Het middenveld is helemaal leeg, op de driehoekige 'turtle' in het midden na. Dat is het tekenvenster, waarin we het resultaat zullen zien van de opdrachten die we geven. Het onderste vakje bevat wat tekst en een vraagteken. Dat is het opdrachtvenster waarin we onze opdrachten zullen tikken.
Het vraagteken in het opdrachtvenster is heel belangrijk, want daarachter hoort een opdracht te staan. Het vraagteken zelf hoef je niet in te tikken. Dat verschijnt vanzelf. Na elke opdracht druk je op de **Enter**-toets.
Op het einde van de vorige aflevering leerden we met een variabele werken. De makers van SuperLogo beelden een variabele uit als een doosje. Je kan er een woord, een getal, een lijst… instoppen. Zodra iets erin zit, kan je het er op elk moment weer uithalen om het in je programma te gebruiken.



Het gemak van variabelen



Figuur 1

Op het einde van de vorige aflevering leerden we voorzichtig met een variabele werken. Maar dat doosje kan veel meer bevatten dan je naam. Je kan er letterlijk elke waarde instoppen. Misschien zie je niet meteen wat je met zo'n waarde kan doen. Laten we een paar praktische voorbeelden geven. We beginnen met een procedure 'driehoek200' aan te maken. Dat doen we als volgt:

```
LEER DRIEHOEK200
 LINKS 90
 VOORUIT 200
 RECHTS 120
 VOORUIT 200
 RECHTS 120
 VOORUIT 200
 RECHTS 210
EIND
```

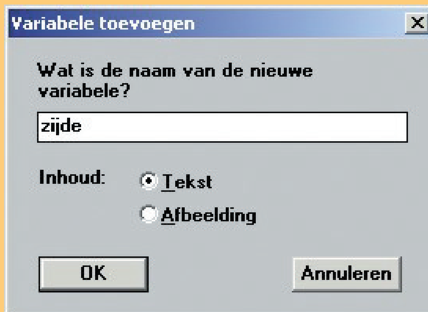
Het resultaat is een gewone gelijkzijdige driehoek en we verkrijgen die door in het opdrachtvenster de procedurenaam 'driehoek200' in te voeren (Figuur 1). Je begrijpt echter dat we voor elke maat driehoek een nieuwe procedure moeten aanmaken: 'driehoek100', 'driehoek150', 'driehoek 250' enzovoort. Dat is natuurlijk onzin. Dan kan je immers net zo goed telkens de hele procedure in de opdrachtlijn intikken. Dat gaat sneller.
Dankzij het gebruik van een variabele kan SuperLogo om het even welke driehoek tekenen met slechts één procedure… en een variabele natuurlijk. De variabele zal dus de lengte van de zijde van de driehoek bepalen en we noemen die dan ook 'zijde'. Weet je nog hoe we een variabele maken? Open het geheugenvenster door op de knop in de taakbalk te klikken of door op de functietoets **F4** te drukken. In het menu **Objecten** kies je voor **Toevoegen variabele...** (kan ook met **CTRL+R**). Daarmee open je het venster **Variabele toevoegen** en daar vullen we de naam 'zijde' in voor deze varia-

kleur en muziek

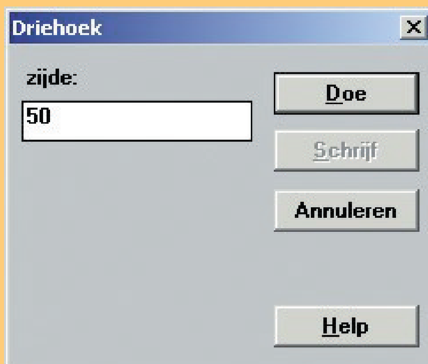
WAT, WAAR EN HOEVEEL?

SuperLogo 2.0 is uitgegeven bij A.W. Bruna Multimedia en kost € 40,75. Het pakket is verkrijgbaar in de boek- en softwarehandel. Meer info vind je op [www.awbruna.nl/superlogo].





Figuur 2



Figuur 3

bele (Figuur 2). Het keuzerondje voor TEKST laten we staan. We willen in dit 'doosje' immers het waardecijfer wegstoppen dat de lengte van een zijde aangeeft. Klik op OK. Je ziet nu de aangemaakte variabele in het lijstje in het geheugenvenster staan. Het doosje is leeg, maar dat hoeft niet lang meer te duren.

We maken nu een nieuwe procedure aan die we gewoon 'driehoek' noemen. Die procedure ziet er zo uit:

```
LEER DRIEHOEK :ZIJDE
 LINKS 90
 HERHAAL 3 [VOORUIT :ZIJDE RECHTS 120]
 RECHTS 90
EIND
```

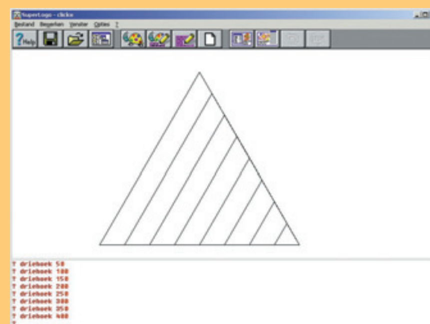
Er zijn wel een paar verschillen met de vorige procedure. Het belangrijkste is dat we na LEER DRIEHOEK ook nog eens :ZIJDE hebben gezet. Dat wil zeggen dat we de procedure driehoek maken met gebruik van de variabele ZIJDE. Let op de dubbelepunt voor ZIJDE. Als je die vergeet, herkent SuperLogo die niet als een variabele.

Vervolgens hebben we eigenlijk precies dezelfde procedure geschreven als voor DRIEHOEK200. Alleen hebben we het cijfer achter de opdracht VOORUIT vervangen door :ZIJDE. We hebben ook wat zuiniger geprogrammeerd door de drie identieke opdrachten tot één samen te vatten die we driemaal herhalen. De opdrachten LINKS 90 en RECHTS 90 dienen alleen maar om de uitgangspositie van de driehoek te herstellen.

Als we nu de opdracht DRIEHOEK geven en op de ENTER-toets drukken, floept het variabelenvenstertje DRIEHOEK open met daarin het vakje ZIJDE (Figuur 3). Vul daar 50 in en klik op de DOE-knop. Meteen wordt je driehoek getekend met een zijde van 50 pixels. In je opdrachtlijn zie je hoe de opdracht DRIEHOEK aangevuld is met het getal 50. Je kan natuurlijk de opdracht ook meteen zelf inbrengen met het getal erbij. Vergeet de spatie niet. Geef achtereenvolgens deze opdrachten:

```
? DRIEHOEK 50
? DRIEHOEK 100
? DRIEHOEK 150
? DRIEHOEK 200
? DRIEHOEK 250
? DRIEHOEK 300
? DRIEHOEK 350
? DRIEHOEK 400
? DRIEHOEK 450
?
```

Je hebt nu negen driehoeken getekend vanuit hetzelfde beginpunt, maar met de zijde telkens vijftig pixels groter (Figuur 4). Je ziet meteen de voordelen van het werken met variabelen. We maken om het even welke driehoek met slechts één procedure en maar één variabele.



Figuur 4

Twee variabelen tegelijk gebruiken

Zou het je ook lukken om de procedure te schrijven om een vierhoek te tekenen, waarbij lengte en breedte verschillen? Denk er eens over na voor je verder leest en probeer het desnoods eens uit.

Lukt het niet meteen? Maak dan deze procedure:

```
LEER VIERHOEK :LENGTE :BREEDTE
 HERHAAL 2 [VOORUIT :LENGTE RECHTS 90
VOORUIT :BREEDTE RECHTS 90]
EIND
```

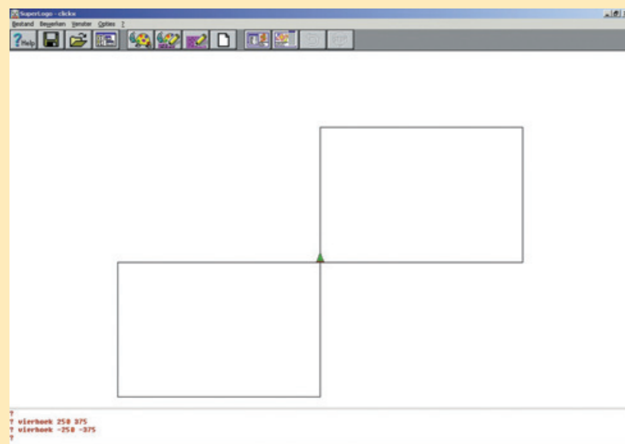
Gesnopen? We geven de opdracht LEER VIERHOEK en we voegen er gewoon de twee variabelen :LENGTE en :BREEDTE aan toe. Dan herhalen we tweemaal VOORUIT met de lengte, RECHTS 90, VOORUIT met de breedte en weer RECHTS 90.

Probeer de procedure uit door de opdracht VIERHOEK te geven. Je ziet nu weer het variabelenvenstertje, maar deze keer kan je beide variabelen invullen: lengte en breedte (Figuur 5). Klik op DOE en je vierhoek wordt getekend.

Is er je niets opgevallen? We hebben helemaal geen variabelen op voorhand aange-



Figuur 5



Figuur 6

maakt! Met de variabele ZIJDE hadden we dat wel gedaan. Dat is dus niet nodig, tenminste als je die variabele alleen maar binnen één procedure gebruikt. Dan maak je ze gewoon tijdelijk aan met de LEER-opdracht.
Terloops nog even vermelden dat je ook negatieve getallen (met een minteken voor) kan opgeven als variabelen. De opdracht VOORUIT -100 is eigenlijk precies hetzelfde als de opdracht ACHTERUIT 100. Dat is wel interessant, want nu kunnen we alleen met de opdracht VOORUIT in onze procedure zowel vooruit als achteruit. Herinner je je nog wat we in een vorige aflevering over coördinaten vertelden? Dan begrijp je meteen ook hoe we volgende tekening gemaakt hebben (Figuur 6).
We deden het zo:

```
? VIERHOEK 250 375
? VIERHOEK -250 -375
?
```

En hoe zou je een vierkant maken?

Een schermvol met één opdracht!

Is het je gelukt een vierkant te maken? Je procedure zou er dan ongeveer zo moeten uitzien:

```
LEER VIERKANT :ZIJDE
 HERHAAL 4 [VOORUIT :ZIJDE RECHTS 90]
EIND
```

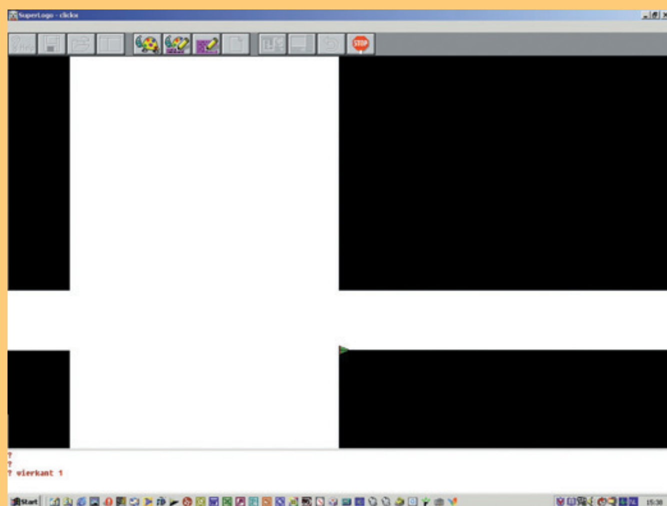
Je ziet het: hier gebruiken we de eerder aangemaakte variabele 'ZIJDE' opnieuw. Je kan nu elk gewenst vierkant maken, zelfs het allerkleinste van slechts één pixel (? VIERKANT 1).
Als je nu achtereenvolgens vierkantjes tekent van telkens één pixel meer, krijg je telkens een iets groter zwart vierkant te zien. Bijvoorbeeld zo:

```
? VIERKANT 1
? VIERKANT 2
? VIERKANT 3
? VIERKANT 4
? VIERKANT 5
? VIERKANT 6
?
```

In feite heb je niets anders gedaan dan telkens een lijntje van één pixel dik langs twee zijden van je vierkant te leggen. Leuk, maar een beetje bewerkelijk. Dat moet makkelijker kunnen. Laten we onze procedure eens als volgt bewerken:

```
LEER VIERKANT :ZIJDE
HERHAAL 4 [VOORUIT :ZIJDE RECHTS 90]
MAAK "ZIJDE :ZIJDE + 1
VIERKANT :ZIJDE
EIND
```

Hier hebben we twee dingen gedaan: nadat het vierkant getekend is, hebben we de variabele ZIJDE een andere waarde gegeven. We hebben als het ware het doosje leeggehaald en er iets anders ingestopt. Dat deden we met de opdracht MAAK "ZIJDE en daarop aansluitend de bestaande variabele + 1 (:ZIJDE + 1). We tellen dus één pixel bij onze variabele waardoor die straks een vierkant gaat tekenen dat één pixel groter is. Vervolgens laten we de procedure zichzelf uitvoeren! Dat doen we door in de procedure de opdracht VIERKANT :ZIJDE te geven. Daardoor wordt het vierkant opnieuw uitgevoerd, maar nu met de nieuwe waarde die we aan de variabele 'zijde' hebben gegeven. Afsluiten en even voorzichtig proberen met ? VIERKANT :ZIJDE.
Maar hemeltje! Wat is dat? Ons hele scherm kleurt zwart. Als het bovenaan rechts vol is, begint het gewoon onderaan opnieuw tot alles zwart is (Figuur 7). En steeds opnieuw worden er nieuwe vierkanten getekend. Dat gaat zo eindeloos verder. Maak er gauw een eind aan door op de functietoets F12 te drukken. In het opdrachtveld verschijnt nu een regel 'Het programma is gestopt in regel 1 van procedure vierkant'. Gelukkig maar.



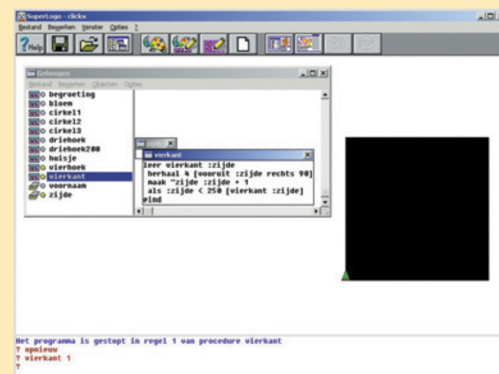
Figuur 7

Stop de wereld! Ik wil eraf!

Zo'n procedure die nooit eindigt, noemt men een lus. Ze blijft zichzelf steeds maar herhalen tot je er nogal bruusk een eind aan maakt met die F12-knop. Zou het dan niet mogelijk zijn om zelf te bepalen hoe vaak zo'n lus herhaald moet worden?
Natuurlijk is dat mogelijk. Je moet gewoon een lus maken waaraan je een voorwaarde verbindt. Zoiets als: teken een telkens groter wordend vierkant tot de zijde 250 pixels groot is. Daarvoor hebben we de handige opdracht ALS. We knutselen nog maar eens aan onze procedure:

```
LEER VIERKANT :ZIJDE
HERHAAL 4 [VOORUIT :ZIJDE RECHTS 90]
MAAK "ZIJDE :ZIJDE + 1
ALS :ZIJDE < 250 [VIERKANT :ZIJDE]
EIND
```

En dat lukt perfect: we tekenen nu een zwartgekleurd vierkant met een zijde van precies 250 pixels (Figuur 8).



Figuur 8

Spelen met lijntjes

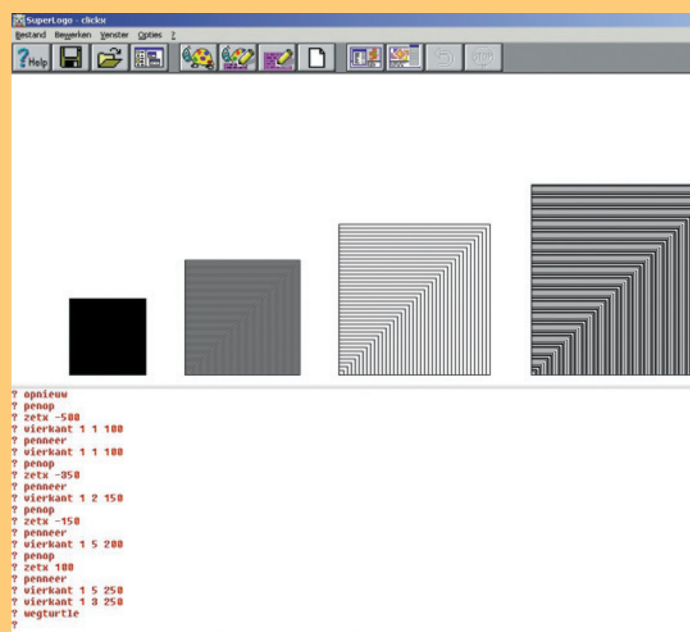
Nu kunnen we een paar leuke dingen doen met ons vierkant. Probeer eens uit wat het geeft als je het aantal pixels met twee verhoogt. Maak daarvoor meteen de nieuwe variabelen **TOENAME** en **GROOTTE** aan, zodat je de grootte later met om het even welk aantal pixels kan laten toenemen zonder in de procedure zelf te moeten knutselen. Onze procedure **VIERKANT** ziet er nu als volgt uit:

```
LEER VIERKANT :ZIJDE :TOENAME :GROOTTE
 HERHAAL 4 [VOORUIT :ZIJDE RECHTS 90]
 MAAK "ZIJDE :ZIJDE + :TOENAME
 ALS :ZIJDE < :GROOTTE [VIERKANT :ZIJDE :
 TOENAME :GROOTTE]
EIND
```

Vergeet vooral niet bij de opdracht **? VIERKANT** ook weer alle variabelen te vermelden. Anders krijg je een foutmelding. Van zo gauw er drie variabelen in een procedure zijn, kan je die niet meer opgeven via een variabelenvenstertje, maar moet je de waarden voor de variabelen in de juiste volgorde in het opdrachtvenster geven: bijvoorbeeld **? VIERKANT 1 5 200**. Met deze procedure kan je tal van prachtige patronen maken. Door telkens de positie van de turtle te veranderen, zetten we er hier vier naast elkaar.

```
? OPNIEUW
? PENOP
? ZETX -500
? PENNEER
? VIERKANT 1 1 100
? PENOP
? ZETX -350
? PENNEER
? VIERKANT 1 2 150
? PENOP
? ZETX -150
? PENNEER
? VIERKANT 1 5 200
? PENOP
? ZETX 100
? PENNEER
? VIERKANT 1 5 250
? VIERKANT 1 3 250
? WEGTURTLE
?
```

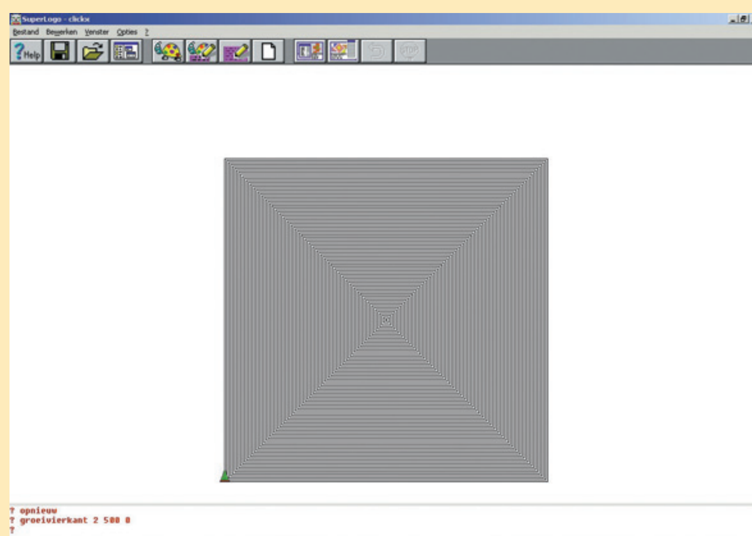
Het resultaat is verbluffend (Figuur 9). Bij het eerste vierkant met zijde=1, toename=1 en grootte=100 krijgen we een aaneengesloten zwart vierkant. Het tweede (1 2 150) lijkt gearceerd in twee verschillende tinten. Het derde (1 5 200) heeft een mooi gelijkmatig lijntjespatroon. Met het vierde is er iets speciaals aan het hand. Let erop dat wij hier tweemaal een even groot vierkant tekenen op dezelfde plaats, maar de ene keer met toename=5, de andere keer met toename=3. Daardoor worden de twee patronen met elkaar verweven. Sommige lijntjes vallen over elkaar, andere vullen de witruimte tussen andere lijnen op. Zo krijg je heel mooie patronen, waarmee je eindeloos kan experimenteren.



Figuur 9

Het groeit langs alle kanten

Nog leuker zou het zijn een vierkant te tekenen dat langs alle zijden tegelijk groeit. Maar hoe leggen we dat aan boord? Eigenlijk is het heel simpel. Kwestie van even de pen op te heffen en te verplaatsen. Maar hoe doen we dat regelmatig en liefst op basis van de reeds gemaakte procedure? We laten de variabele **TOENAME** weg en maken een nieuwe die we **POSITIE** noemen. We schrijven nu de nieuwe procedure **GROEIVIERKANT** als volgt:



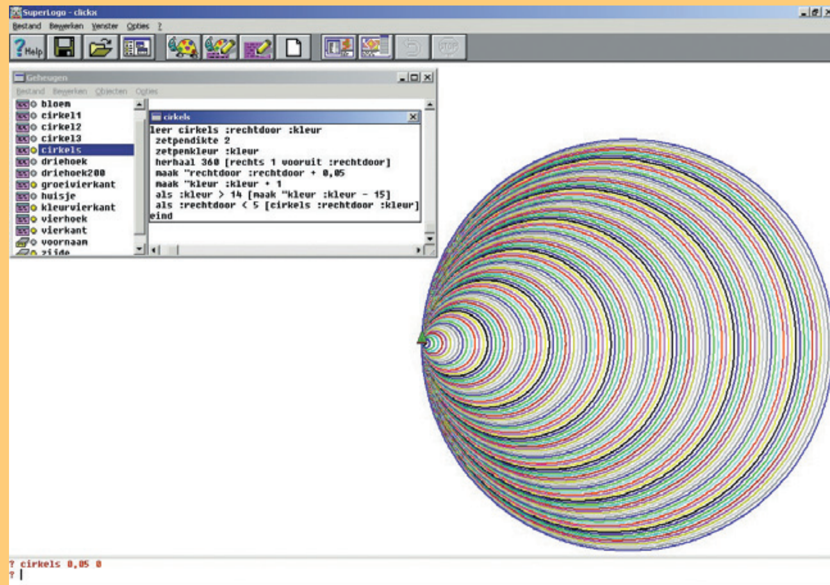
Figuur 10

```
LEER GROEIVIERKANT :ZIJDE :GROOTTE :POSITIE
HERHAAL 4 [VOORUIT :ZIJDE RECHTS 90]
MAAK "ZIJDE :ZIJDE + 6
MAAK "POSITIE :POSITIE - 3
ALS :ZIJDE < :GROOTTE [PENOP ZETX :POSITIE
ZETY :POSITIE PENNEER GROEIVIERKANT :ZIJDE
:GROOTTE :POSITIE]
EIND
```

Als we de procedure uitvoeren als '**? GROEIVIERKANT 2 500 0**', krijgen we een bijna schermvullend resultaat, een vierkant dat vanuit het midden van de pagina groeit (Figuur 10). Telkens wordt de zijde met zes pixels groter gemaakt, maar de positie, zowel van x als van y verminderen we met drie pixels. In de voorwaardelijke opdracht **ALS** heffen we eerst onze pen op en veranderen dan de posities van x en van y. Dan zetten we onze pen weer neer en voeren opnieuw het groeivierkant uit. Als je start met positie 0, groeit je vierkant vanuit het midden van je scherm. Vul je een andere waarde in, dan zal daar het middelpunt van het vierkant liggen.

Een kleurige kegel

Je kan nog veel leukere dingen doen met variabelen. Herinner je je nog onze procedure CIRKEL1? Laten we die eens als basis gebruiken om een kleurige kegel te tekenen. We maken de procedure CIRKELS aan met twee variabelen: RECHTDOOR en KLEUR. Met RECHTDOOR bedoelen we het getal achter de opdracht VOORUIT in onze basiscirkel. Die laten we telkens met 0,05 pixels toenemen na elke cirkel.
Om de kleur te veranderen, voeren we de variabele KLEUR in. Je weet dat de opdracht ZETPENKLEUR met de cijfers '0' tot en met '15' met de zestien basiskleuren kan uitgerust worden. We willen de kleur bij elke cirkel veranderen. We moeten er daarbij wel voor zorgen niet boven het getal vijftien uit te komen, want de opdracht ZETPENKLEUR kan met hogere enkelvoudige getallen niet overweg. Sterker nog: we willen ons zelfs beperken tot het getal veertien. De reden daarvoor is eenvoudig: de kleur '15' is namelijk wit. Vermits onze achtergrond wit is, kunnen we die beter niet gebruiken. Wit tekenen op wit papier is niet echt aangewezen. Als het kleurgetal dus groter wordt dan veertien, trekken we er vijftien af, waardoor we weer op 0 (nul) uitkomen, de kleur zwart. Zo beginnen we weer van voren af aan.



Figuur 11

We dragen er ook zorg voor dat de waarde van de variabele RECHTDOOR niet boven de vijf pixels stijgt, want dan wordt onze kegel groter dan ons scherm en dat is niet de bedoeling. Hoe we dat weten? Gewoon uitproberen.
De procedure CIRKELS ziet er dus zo uit:

```
LEER CIRKELS :RECHTDOOR :KLEUR
ZETPENDIKTE 2
ZETPENKLEUR :KLEUR
HERHAAL 360 [RECHTS 1 VOORUIT :RECHTDOOR]
MAAK "RECHTDOOR :RECHTDOOR + 0,05
MAAK "KLEUR :KLEUR + 1
ALS :KLEUR > 14 [MAAK "KLEUR :KLEUR - 15]
ALS :RECHTDOOR < 5 [CIRKELS :RECHTDOOR :KLEUR]
EIND
```

Alles begrepen? Dan voeren we de procedure cirkels uit met volgende argumenten: '? CIRKELS 0,05 0'. We zien onze tekening groeien en het resultaat mag er zijn (Figuur 11).

Met een streepje muziek

Je kan met SuperLogo ook muziek maken. Maak maar eens een procedure die je BROEDERJACOB noemt. Je schrijft ze zo:

```
LEER BROEDERJACOB
SPEELTOON [C D E C C D E C E F 2G E F 2G 8G 8A 8G 8F E C 8G 8A 8G 8F E C C OB G O# 2C C OB G O# 2C]
EIND
```

Voer nu de procedure uit door '? BROEDERJACOB' in de opdrachtregel te tikken. Klinkt het zo'n beetje?
De opdracht SPEELTOON wordt gevolgd door een zogenaamde 'lijst' met noten. De eerste acht zal je misschien wel kunnen ontcijferen. Het zijn de notenletters: een 'c' staat voor do, een 'd' voor re, een 'e' voor mi, een 'f' voor fa, een 'g' voor sol, een 'a' voor la en een 'b' voor si. Je kan de lengte van een noot bepalen door er een cijfer 1, 2, 4, 8, 16 of 32 voor te plaatsen. Die staan voor een hele, halve, kwart, achtste, zestiende of tweeëndertigste noot. Met de hoofdletter 'L' kan je de standaardlengte voor alle volgende noten veranderen. Standaard staat SuperLogo op L4 en zijn alle niet nader bepaalde noten, kwartnoten.
Een verhoogde noot (kruis) krijgt een hekje na de noot (#), een verlaagde noot (b-mol) een b erna. Zet je een punt na de noot, dan duurt die anderhalve keer zo lang. Een rust kan natuurlijk ook. Die bestaat uit de hoofdletter P. Standaard is P een kwartrust, maar net als bij de noten kan je ook hier met de nodige cijfers de rust verlengen of verkorten.
Een hoofdletter O gevolgd door een hekje (O#) doet je in een hoger octaaf belanden. Een hoge do wordt dus 'O# c' en je blijft in dat hoge octaaf tot er weer een 'Ob' volgt. Je kan hier ook een octaafnummer voor gebruiken. De octaven zijn dan genummerd van 0 tot 6 en worden weergegeven als O0, O1, O2 enz. Standaard staat SuperLogo op O4.
Het tempo van je stuk kan je met de letter 'T' een metronoomwaarde meegeven. Standaard staat SuperLogo op T120.
Wil je al je instellingen weer ongedaan maken en opnieuw de standaardwaarden instellen, zet dan de hoofdletter 'R' in je lijst. En nu maar experimenteren. Bij wijze van afsluiter geven we je nog een melodietje mee.

```
LEER FURELISE
SPEELTOON [T80 O4 L16 E D# E D# E OB B O# D C OB 8A 16P C E A 8B 16P E G# B O# 8C 16P OB E O# E D# E D# E OB B O# D C OB 8A 16P C E A 8B 16P E O# C OB B 8A 8P]
SPEELTOON [T80 O4 L16 E D# E D# E OB B O# D C OB 8A 16P C E A 8B 16P E G# B O# 8C 16P OB E O# E D# E D# E OB B O# D C OB 8A 16P C E A 8B 16P E O# C OB B 8A 16P]
SPEELTOON [T80 O4 L16 OB B O# C D 8E. OB G O# F E 8D. OB F O# E D 8C. OB E O# D C OB 8B 16P E O# E 16P 16P E O# E 16P 16P OB D# E 16P 16P]
SPEELTOON [T80 O4 L16 D# E D# E D# E OB B O# D C OB 8A 16P C E A 8B 16P E G# B O# 8C 16P OB E O# E D# E D# E OB B O# D C OB 8A 16P C E A 8B 16P E O# C OB B 8A 8P]
EIND
```

Voer de procedure FURELISE uit en je hoort een stukje van de Für Elise van Ludwig van Beethoven. Gelukt? Vergeet dan je project niet te bewaren.

— Willy Schuyesmans —